3С

УДК 004.896

С.С. Синельников
Государственный университет информатики и искусственного интеллекта, г. Донецк, Украина

Анализ характеристик интеллектуальности алгоритмов поиска и классификация методов поиска

Рассмотрены требования к интеллектуальным методам поиска данных в массиве, проведен теоретический анализ методов поиска, дана классификация методов поиска.

Введение

Базы данных интеллектуальных систем представляют собой, как правило, набор таблиц, удовлетворяющих 3-й нормальной форме [1]. Таблицы являются хранилищем данных, которые проиндексированы и упорядочены. Фактически таблицы являются массивами с отсортированными данными. Эффективность такой организации хранения данных определяется реализацией команд языка выборки данных (SQL) таких, как: прямое вхождение (inner join), диапазон (between), перечисление (in), поиск и несовпадение (=, <>), больше и меньше (>, >=, <, <=) и другие. Стремление разработчиков СУБД к повышению скорости выполнения этих команд позволяет говорить об актуальности задачи поиска данных, которая является основополагающей для всех команд SQL.

Задача поиска [2-4] заключается в нахождении индекса i элемента массива m по заранее известному значению элемента $= key$, то есть поиск индекса i, при котором $m[i] = key$.

Решение задачи поиска в массиве возможно линейным перебором, бинарным методом, интерполяционным методом, с применением хеширования, численных методов [5]. Каждый из методов имеет свои достоинства и недостатки, хорошо работают для определенных данных и в конкретных случаях, что не позволяет говорить об их общности и интеллектуальности. В данной работе вводятся требования, которым должны удовлетворять методы поиска с интеллектуальными характеристиками. Впервые предложена классификация методов поиска, которая обобщает работу методов для отсортированных, не отсортированных и частично упорядоченных данных.

1 Классификация методов поиска в зависимости от свойств хранимых данных

Данные в массиве m размера N могут располагаться друг относительно друга тремя способами:

- данные не отсортированы;
- данные частично отсортированы;
- данные отсортированы (по возрастанию или убыванию).

1.1 Методы поиска для не отсортированных данных

В случае, когда данные не отсортированы, имеется возможность применения двух методов поиска: линейный поиск и хеширование.

Линейный поиск – обычный перебор элементов для сравнения с ключом. Средняя скорость работы данного метода – N/2, что позволяет говорить о скорости линейного порядка – N. Метод является самым медленным из всех известных и эффективен в том случае, если N <= 3. При N = 3 количество сравнений с ключом в среднем совпадает с бинарным методом, который к тому же использует дополнительные просчеты.

Поэтому в СУБД линейный метод поиска применяется, если количество элементов не превышает 3, что на практике встречается крайне редко.

Хеширование – процесс подбора функции f, которая для массива m и ключа key удовлетворяла бы следующим свойствам:

$$\begin{cases} m[i] = key \\ f(key) = i \end{cases}.$$

Как правило, на практике используют функцию f вида

$$f(key) = g(key)\%N,$$

в которой остаток от деления (%) на N позволяет не выходить за пределы массива (индексация начинается с нуля), а некая функция g обеспечивает равномерный разброс элементов по массиву.

Наибольшая трудность для данного метода – это подбор функции g, который осуществляет в большинстве случаев человек, что и определяет слабую обобщенность данного метода. При выборе достаточно эффективной функции хеширования скорость поиска может быть одной из самых высоких по сравнению с другими методами и достигать в наилучшем случае одного сравнения. Хеширование, несмотря на свои достоинства, имеет ряд недостатков: вычислительная сложность метода зависит от хеш-функции; метод недостаточно обобщен для различных данных; появление цепочек данных с одинаковым значением хеш-функции. Но главный недостаток – это невозможность использования принципов хеширования для реализации операций «>,>=,<,<=», выбор из диапазона. Безусловно, это сужает круг задач, в которых можно использовать хеширование. Поэтому в СУБД, как правило, не всегда применяется хеширование, а если применяется, то как вспомогательный элемент основного метода поиска.

1.2 Методы поиска для частично отсортированных данных

В случае, когда данные частично отсортированы, как будет показано ниже, возможно применение градиентного метода поиска или методов, основанных на поиске минимума (или нуля) таблично заданной функции.

Для доказательства возможности применения градиентного метода к задаче поиска данных в частично отсортированном массиве вычтем из всех элементов массива искомый элемент key и возьмем модуль полученной величины. Очевидно, что в таком случае задача сводится к поиску минимума таблично заданной функции вида:

$$|m[i] - key|.$$

В точке с индексом i, при котором m[i] – key = 0, находится искомый элемент. Так как функция может иметь не один минимум, то градиентный метод должен обеспечивать возможность выхода из локального минимума.

Для поиска требуемого элемента может показаться, что необходимо выполнять операцию вычитания ключа и получения модуля для всех элементов. На практике эта процедура выполняется только при необходимости обращения к элементу массива и производится только для него.

Градиентный метод в классическом виде, который применяется для непрерывных функций, а не для дискретных, необходимо модифицировать и адаптировать для рассматриваемой задачи поиска. Но, тем не менее, данный метод является наиболее общим для данных с различными свойствами как отсортированных, так и не отсортированных, что обусловливает его универсальность. Градиентный метод использует в вычислении нового индекса значение производной. Известно, что производная для непрерывной функции f есть предел отношения разности приращения функции к приращению аргумента x при стремлении приращения аргумента к нулю

$$f'(x) = \lim_{\Delta x \to 0} \frac{\mathrm{f}(\mathrm{x} + \Delta x) - \mathrm{f}(\mathrm{x})}{\Delta x}.$$

Для дискретного случая получаем конечные разности:

$$\Delta m[i] = \mathrm{m}[\mathrm{i}+1] - \mathrm{m}[\mathrm{i}].$$

В дальнейшем под производной дискретной функции будем понимать вышеописанную трактовку. Тогда итерационная формула получения нового индекса в соответствии с градиентным методом примет вид

$$\mathrm{i}_{\mathrm{n}+1} = \mathrm{i}_{\mathrm{n}} - \alpha \cdot \Delta\mathrm{m}[\mathrm{i}_{\mathrm{n}}],$$

где α – величина шага, которая может меняться на каждой итерации.

Выбор α составляет наибольшую трудность для данного метода.

1.3 Методы поиска для отсортированных данных

Для отсортированных данных возможно применение целого ряда методов, которые основаны на численных методах поиска нуля таблично заданной функции. К таковым относятся: бинарный метод, метод хорд (интерполяционный метод), метод итераций и метод Ньютона (касательных).

Действительно, если вычесть из элементов массива ключ, то задача сводится к поиску нуля таблично заданной функции:

$$\mathrm{m}[\mathrm{i}] - \mathrm{key}.$$

Заметим, что вычитание ключа выполняется только при необходимости обращения к элементу массива и производится только для него.

Для всех численных методов требуется выполнение необходимого условия их применимости: это – знакопостоянство первой производной, то есть данные должны быть упорядочены (по возрастанию или убыванию), что для отсортированных данных естественно выполняется.

Каждый из вышеперечисленных методов требует дополнительных условий работы, поэтому применение этих методов должно проходить строго с их учетом.

На данный момент методы поиска, основанные на численных методах, имеют наибольшую скорость работы по сравнению с вышеописанными, но они требуют выполнения дополнительных условий работы, что сказывается на их обобщенности. Тем не менее, практически все СУБД используют бинарный метод поиска, так как легче поддерживать условия хранения данных (их сортировка).

В работе [6] показаны условия применимости и работы итерационного метода, метода хорд и Ньютона. Бинарный метод использует в качестве разделяющего средний элемент из диапазона поиска: $\text{middle} = \frac{low + high}{2}$.

Метод хорд использует в качестве разделяющего элемент с индексом, полученным как пересечение оси индекса с прямой, проведенной через граничные точки диапазона поиска:

$$\text{middle} = low - \frac{(\text{m[low]} - \text{key}) \cdot (\text{high} - \text{low})}{\text{m[high]} - \text{m[low]}}.$$

Метод хорд и бинарный метод не требуют дополнительных условий работы.

Метод итераций использует следующую рекуррентную формулу для получения индекса искомого элемента:

$$\text{I}_{\text{n+1}} = \text{I}_{\text{n}} + (\text{key} - \text{m}[\text{I}_{\text{n}}]) \cdot \text{C},$$

где $\text{C} = \frac{1}{\max\{|\text{m[I+1]} - \text{m[I]}|,\ \text{I} = \overline{0, N-2}\}}$.

Метод итераций требует расчета множителя C до его применения и не требует дополнительных условий работы. Наибольший плюс данного метода в том, что он использует однонаправленное движение по набору данных для поиска элемента, что может быть полезно для списков и поиска в файлах.

Метод хорд и метод итераций наиболее эффективны, когда данные образуют арифметическую прогрессию или последовательность, близкую к ней. Также эти методы чувствительны к резким скачкам между рядом стоящими элементами, которые отрицательно сказываются на скорости поиска.

Метод Ньютона (касательных) можно применять, если первая и вторая производные m[i] знакопостоянны, то есть для отсортированного по возрастанию массива:

$$\begin{cases} \text{m[i]} \le \text{m[i+1]} \\ \text{m[i+1]} - \text{m[i]} \le \text{m[i+2]} - \text{m[i+1]}. \end{cases}$$

Метод Ньютона (касательных) использует следующую рекуррентную формулу для получения индекса искомого элемента:

$$\text{I}_{\text{n+1}} = \text{I}_{\text{n}} - \frac{\text{m}[\text{I}_{\text{n}}] - \text{key}}{\text{m}[\text{I}_{\text{n}}] - \text{m}[\text{I}_{\text{n}} - 1]}.$$

Аналогично можно получить рекуррентную формулу для условий:

$$\begin{cases} \text{m[i]} \le \text{m[i+1]} \\ \text{m[i+1]} - \text{m[i]} \ge \text{m[i+2]} - \text{m[i+1]}. \end{cases}$$

Метод Ньютона быстрее других методов, но узко направлен и в общем случае не применим. Наиболее быстрым и обобщенным считается бинарный метод поиска, так как он не использует дополнительных вычислений, стабилен в скорости для любого набора данных, прост в реализации. Поэтому он применяется в большинстве СУБД.

2 Анализ характеристик алгоритмов поиска и требований к методам поиска для повышения их интеллектуальности

Опираясь на предложенную выше классификацию методов поиска проведем анализ характеристик алгоритмов поиска и требований к методам поиска для повышения их интеллектуальности. Эффективность метода поиска зависит от того,

на сколько он быстро решает поставленную задачу, учитывает известные особенности хранения данных, а также свойства хранимой информации. Таким образом, под интеллектуальностью метода будем понимать его возможность использовать дополнительную информацию о данных.

К такой дополнительной информации можно отнести следующие особенности данных, которые заранее известны:

1) как много копий в хранимых данных или сколько различных элементов;

2) искомый элемент однозначно присутствует или отсутствует в массиве;

3) особенности данных, носящие функциональный характер или свойства взаимного расположения элементов.

Первая особенность данных напрямую показывает, с какой вероятностью возможно найти искомый элемент при первом же сравнении данных с ключом на равенство. Поэтому эта особенность дает информацию о том, как часто использовать операцию сравнения «равенство» (=), и ее необходимо учитывать. Вторая особенность позволяет сделать вывод о том, стоит ли вообще использовать «равенство» (=) для сравнения данных с ключом. Если элемент присутствует, то использование операции «равенства» оправдано, иначе – нет (например, для поиска места вставки (insert) уместно использовать операции «>,<,>=,<=»). Третья особенность позволяет использовать различные методы поиска, основанные на численных методах, которые учитывают функциональные зависимости между данными.

Интеллектуальные методы должны использовать эту информацию для более быстрого поиска и соответственно удовлетворять следующим требованиям:

– если известна дополнительная информация о данных, то интеллектуальный метод поиска должен ее учитывать;
– если известна дополнительная информация об искомом элементе, то интеллектуальный метод поиска должен ее учитывать;
– если нет дополнительной информации о данных и элементе, то интеллектуальный метод поиска должен это учитывать;
– если есть возможность анализа данных, то интеллектуальный метод должен ее производить до его применения поиска.

Все необходимые действия для поиска очень сходны с интеллектуальной деятельностью человека, так как перед тем, как что-либо искать, он анализирует предмет поиска, хранимые данные и их свойства. Поэтому методы поиска, обладающие сходными свойствами с методами поиска человека, можно называть интеллектуальными, или с элементами интеллекта. К таковым возможно отнесение методов поиска, которые используют стратегию поиска, сходную со стратегией человека, основанную на анализе размера массива:

– если размер массива достаточно велик, то уместно делать проверку на отсутствие ключа в массиве;
– если размер средний, возможно присутствие копий, то элемент, скорее присутствует в массиве и проверку на отсутствие элемента следует убрать, а добавить проверку на совпадение с ключом;
– если размер массива мал, то просто просмотрим все элементы для поиска ключа.

Методы поиска должны быть достаточно адаптивны к требованиям интеллектуальности, то есть в интеллектуальном методе должна быть возможность легко добавить еще какой-либо метод, который лучше работает с массивом определенного размера. Также должна быть возможность убрать какой-либо метод, если заранее

известен максимальный размер массива. Эта гибкость позволяет говорить об интеллектуальности метода и СУБД, которая его использует. Настраивая метод, СУБД может получить прирост в скорости работы для массивов с различными размерами.

Выводы

В данной работе предложены и проанализированы требования, которым должны удовлетворять методы поиска с элементами интеллекта. Все требования были взяты на основе вероятности выполнения различных операций сравнения в зависимости от хранимых и искомых данных. Показана связь задачи с методами теории вероятности. Выявлено, что такой подход эффективен для задачи поиска с дополнительными условиями. Впервые предложена классификация методов поиска в зависимости от свойств взаимного расположения элементов, которая обобщает работу методов для отсортированных, не отсортированных и частично отсортированных данных.

Литература

1. Коннолли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика. – М.: Вильямс, 2000. – 1120 с.
2. Алгоритмы: построение и анализ / Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. – 2-е изд. – М.: Вильямс, 2005. – 1296 с.
3. Кнут Д. Искусство программирования. – 3-е изд. – М.: Вильямс, 2005. – 720 с.
4. Седжвик Р. Фундаментальные алгоритмы на С++: Ч. 1 – 4 // Анализ, структуры данных, сортировка, поиск: Пер. с англ. – Diasoft. – 2001. – 687 с.
5. Вержбицкий В.М. Численные методы. – М: Высшая школа, 2001. – 382 с.
6. Синельников С.С. Применение численных методов к задаче поиска данных в отсортированном массиве // Радіоелектронні і комп'ютерні системи. – 2007. – № 1. – С. 68-73.
7. Страуструп Б. Язык программирования С++. – М: Бином, 2005. – 1098 с.

С.С. Синельников

Аналіз характеристик інтелектуальності алгоритмів пошуку та класифікація методів пошуку
Розглянуто вимоги до інтелектуальних методів пошуку даних у масиві, проведено теоретичний аналіз методів пошуку, дана класифікація методів пошуку.

Статья поступила в редакцию 17.04.2008.